AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method for executing program instructions comprising the steps of:

assigning each individual ones of a plurality of program steps a unique number;

issuing a program step to an execution queue out of program order;

selecting a specific one of a plurality of ~~numbered~~ locations in the execution queue based upon the unique number of the issued program step, each ~~numbered~~ location having an instruction valid bit and the execution queue having a certain total number of ~~numbered~~ locations;

determining the value of the instruction valid bit associated with the selected ~~numbered~~ location;

~~using the certain total number and~~ based on the unique number of the issued program step, calculating value of a status bit for the selected ~~numbered~~ location; and

based upon the determined value of the instruction ~~value~~ valid bit and the calculated value of the status bit, determining availability of the selected ~~numbered~~ location such that the issued program step is stored in the selected ~~numbered~~ location if the selected ~~numbered~~ location is available, and ~~a full flag for~~ issuing an indication that the execution queue is ~~issued~~ full if the selected ~~numbered~~ location is not available.

2. (Currently Amended) The method of claim 1, wherein the unique ~~number is a~~ numbers are monotonically ascending series of integers assigned to the program steps in the same order as the program is executed.

3. (Original) The method of claim 1, wherein the execution queue is one of a load queue and a store queue.

4. (Currently Amended) The method of claim 1, wherein the program step is issued to the execution queue in an order determined by an availability of a selected one of a plurality of computation resources, the determined order different than the program order.

5. – 6. (Cancelled)

7. (Currently Amended) The method of claim [[6]] 21, wherein the divisor of the modulus ~~calculation~~ operation is equal to [[a]] the queue entry number ~~of locations in the queue~~.

8. (Currently Amended) The method of claim 7, ~~wherein~~ further comprising switching the status bit of the selected ~~numbered~~ location ~~is switched~~ when [[said]] the selected location becomes invalid.

9. (Cancelled)

10. (Currently Amended) An apparatus for executing program instructions comprising:

means for assigning each individual ones of a plurality of program steps a unique number;

means for issuing a program step to an execution queue out of a program order;

means for selecting a specific one of a plurality of ~~numbered~~ locations in the execution queue based upon the unique number of the issued program step, each ~~numbered~~ location having an instruction valid bit and the execution queue having a total number of ~~numbered~~ locations;

means for determining the value of the instruction valid bit associated with the selected ~~numbered~~ location;

means for ~~using the total number and the unique number of the issued program step to calculate~~ calculating a value of a status bit based on the unique number for the selected ~~numbered~~ location; [[and]]

based upon the determined value of the instruction valid bit and the calculated value of the status bit, means for determining availability of the selected ~~numbered~~ location in the execution queue~~, such that~~;

means for storing the issued program step ~~is stored~~ in the selected ~~numbered~~ location if the selected ~~numbered~~ location is available[[,]]; and ~~a full flag~~

means for issuing an indication that [[for]] the execution queue is ~~issued~~ full if the selected numbered location is not available.

11. (Cancelled)

12. (Currently Amended) The apparatus of claim 10 wherein the unique ~~number is a~~ numbers are monotonically ascending series of integers assigned to the program steps in the same order as the program is executed.

13. (Original) The apparatus of claim 10, wherein the execution queue is one of a load queue and a store queue.

14. (Currently Amended) The apparatus of claim 10, wherein the execution queue is selected out of a plurality of execution queues based on the type of the issued program step, and the program step is issued to the execution queue in an order determined by an availability of a selected one of a plurality of computation resources, the determined order different than the program order.

15. - 16. (Cancelled)

17. (Currently Amended) The apparatus of claim [[16]] 28, wherein the divisor of the modulus ~~calculation~~ operation is equal to [[a]] the queue entry number ~~of locations in the queue~~.

18. (Currently Amended) The apparatus of claim 17, wherein the status bit of the selected ~~numbered~~ location is switched when said location becomes invalid.

19. (Cancelled)

20. (New) The method of claim 1, further comprising:

performing a numeric operation on the unique number of the issued program step to calculate a queue entry number that selects the specific one of the plurality of locations in the execution queue.

21. (New) The method of claim 20, wherein performing the numeric operation comprises performing a modulus operation on the unique number to calculate the queue entry number that selects the specific one of the plurality of locations in the execution queue.

22. (New) The method of claim 21, wherein performing the modulus operation also produces the value of the status bit.

23. (New) The method of claim 20, wherein performing the numeric operation also produces the value of the status bit.

24. (New) The method of claim 20, wherein each location of the execution queue further is associated with a respective stored status bit, and wherein determining availability of the selected location is based on the determined value of the instruction valid bit and on comparing the calculated value of the status bit with the stored status bit associated with the selected location.

25. (New) The method of claim 24, wherein the selected location is determined to be unavailable in response to determining that the stored status bit associated with the selected location does not match the calculated status bit.

26. (New) The method of claim 25, further comprising changing a state of the stored status bit of each of the plurality of locations of the execution queue in response to completion of all program steps stored in the plurality of locations.

27. (New) The apparatus of claim 10, further comprising means for performing a modulus operation on the unique number of the issued program step to calculate a queue entry number that selects the specific one of the plurality of locations in the execution queue.

28. (New) The apparatus of claim 27, wherein performing the modulus operation also produces the value of the status bit.

29. (New) The apparatus of claim 28, wherein each location of the execution queue further is associated with a respective stored status bit, and wherein determining availability of the selected location is based on the determined value of the instruction valid bit and on comparing the calculated value of the status bit with the stored status bit associated with the selected location.

30. (New) The apparatus of claim 29, further comprising means for changing a state of the stored status bit of each of the plurality of locations of the execution queue in response to completion of all program steps stored in the plurality of locations.

31. (New) An apparatus for executing program instructions comprising:

a first queue to store program steps in a program order, the program steps assigned respective program numbers that correspond to the program order;

an execution queue having a plurality of locations,

the first queue to transfer at least some of the program steps to the execution queue out of the program order,

the locations of the execution queue to store respective program steps based on queue entry numbers calculated from numeric operations on the program numbers of respective program steps,

each location of the execution queue associated with an instruction valid bit for indicating whether the respective location is available, each location of the execution queue further associated with a stored status bit, the first queue to further determine whether a particular location of the execution queue is available by:

calculating a status bit based on the program number of the respective program step; and

comparing the calculated status bit with the stored status bit to determine whether the particular location is available.